INVENTION TITLE

System and Method to distribute reasoning and pattern matching in forward and backward chaining rule engines

DESCRIPTION

Heading

Summary

[Para 1] The invention solves performance issues previously ignored by researchers in the field of artificial intelligence. Using existing techniques, a rule engine implementing Dr. Forgy's original RETE algorithm cannot reason over extremely large datasets or partitioned data. The invention solves the following limitations.

- Dividing the pattern matching between multiple rule engines and sharing the partial and complete matches efficiently.
- Efficiently dividing the pattern matching across several rule engines dynamically.
- Efficiently distributing the indexes of the working memory across multiple machines.
- Efficiently distributing one or more conditional elements of a rule to remote systems and dividing the work across a cluster of computer systems.

[Para 2] In generally, one form of the invention is a computer program which implements the RETE extension to perform distributed reasoning. The device would receive input from an external source and derive a conclusion based a set of rules. The rule defines concrete actions, which should be executed when all the conditions are satisfied. The device can be used in a variety of applications like order management, regulatory compliance, military command control applications or business process automation. The invention is particularly well suited to large real-time systems like military command control systems or security trading systems, which partition data across several systems in multiple physical locations.

Detailed Description of the preferred embodiment

[Para 3] Distributed reasoning, unlike collaborative agents provides methods by which multiple rule engines reason over large datasets in real-time. Whereas collaborative agents use agents to reason over discrete problems, it cannot reason over large sets of data. Furthermore, collaborative agents" techniques require the rule engine to load all necessary data within the same engine. For small datasets, these techniques prove to be powerful, but they do not scale for large datasets.

[Para 4] A forward chaining rule engine utilizing RETE algorithm, can reason over large datasets when nodes are distributed to other systems. This allows the system to match on a specific instance of a class (also called business object) without requiring the object instance reside in the same working memory and be locally available. It is important to note the engine will reason over shared data resident in other engines using data steams or indexes. This reduces the memory requirements and improves efficiency. It also enables the engines to share all or part of their reasoning network and distribute the process. Distributing nodes of a set of rules across multiple systems allows each engine to perform pattern matching on a single object instance and route the results to the originating system. Unlike load balancing techniques, a true distributed reasoning system does not require all systems of a cluster to deploy identical sets of rules, which creates redundant rules within the environment and increases maintenance costs. In a distributed reasoning/distributed pattern matching system, each system deploys a different set of rules (also called module or rule set), based on its own needs and configuration requirements. At runtime, the rule engines monitor resource utilization and distribute nodes dynamically and on demand.

[Para 5] Prior techniques relied on load balancing techniques to prioritize and categorize atomic processes. This approach requires every system to have all required data locally, leading to multiple data caches. In a production environment with large datasets, each

system cannot load all required data and data synchronization becomes a potential problem. Prior research into multi-threaded rule engines were unsuccessful because of the cost of maintaining locks. Distributed reasoning does not use locking mechanism and does not suffer from synchronization issues.

Remarks

[Para 6] I have amended the claims in accordance to the instructions provided by the patent examiner. The invention described in the patent was previously impractical due to the processing power of personal computers. Researchers in the field of Artificial intelligence did not consider these techniques and have not considered them. The invention is currently feasible due to advances in the processing power of personal computers and high-speed networks. With this invention, it is possible to build large distributed reasoning systems made of hundreds and thousands of personal computers. By distributing the indexes and facts between thousands of systems effectively, a forward and backward chaining rule engine can reason over hundreds of millions of facts using inexpensive hardware. Without this invention, the only other technique for reasoning over large datasets is to get larger mainframes with more memory and processing power. As research has shown, scaling up the hardware rapidly reaches a limit and cannot scale to the same level as a distributed reasoning system.

What is claimed is:

[Claim 1] (currently amended) A rule engine which implements the RETE algorithm with novel extensions, which can distribute patterns defined by a set of rules ~~support distributed pattern matching~~.

[Claim 2] (currently amended) A system according to claim 1 converts an object structure to an internal structure, which the rule engine uses to perform operations and reason over data ~~into un-ordered facts within the engine~~.

[Claim 3] (currently amended) A system according to claim 1 stores data in a working memory; which may span multiple partitions across several computer systems and is capable of distributing indexes across multiple rule engines ~~implements the notion of working memory as described in reference article 1~~.

[Claim 4] (currently amended) A system according to claim 1 creates an input nodes for each class type and is capable of propagating data through a local network, or remotely to a rule engine running on a separate computer system ~~Input nodes as defined in reference article 1 are extended with the capability to route objects to a remote rule engine or use it locally~~.

[Claim 5] (currently amended) A system according to claim 1 uses a 2 input Join node to evaluate 2 or more data elements and determines if the data should propagate further through the network on a separate system or locally ~~Join nodes as defined in reference article 1 are extended with the capability to route matched patterns locally or remotely~~.

[Claim 6] (currently amended) A system according to claim 1 uses a Terminal nodes to indicate all the conditions of a rule have been met and is a candidate for immediate or delayed execution locally or on a remote computer system ~~Terminal nodes as defined in reference article 1 are extended with the capability to route matched patterns locally or remotely~~.

[Claim 7] (currently amended) A pattern for a given class type ~~template~~ according to claim 2 uses a linear sequence of nodes representing the patterns to match for the class ~~an object~~.

[Claim 8] (currently amended) A system according to claim 1 implements an abstraction layer for retrieving remote facts residing in a separate rule engine process.

[Claim 9] (original) A system according to claim 1 uses a call back mechanism between the working memory, the rule engine and the objects; which objects use to notify the rule engine of changes.

[Claim 10] (currently amended) A system according to claim 1 requires object instantiations to implement a ~~base~~ defined interface for the call back mechanism, which may be added to the classes using byte code manipulation.

[Claim 11] (currently amended) A system according to claim 1 monitors the resource usage; which is used to determine when RETE nodes are distribute to other rule engines.

[Claim 12] (original) A system according to claim 1 uses rules to manage the distribution of nodes to remote systems.

[Claim 13] (currently amended) A system according to claim 1 distributes pattern matching by ~~serializing~~ copying the nodes and sending them to a remote system; which are then added to remote engine's network.

[Claim 14] (currently amended) A system according to claim 1 distributes the class type node ~~input~~, 2 input node ~~join~~, terminal node and intra-element nodes to a remote system.

[Claim 15] (original) Distributed nodes according to claim 14 maintains a list of remote systems which depend on the indexes produced by pattern matching.

[Claim 16] (currently amended) A system according to claim 1 will serialize the values of an object to a remote system, if the corresponding pattern matches against a remote object pattern, or it may perform join using the indexes directly.

[Claim 17] (currently amended) A system according to claim 1 may serialize the data object and its values to a remote system, if the object contains procedural logic and functional attachments ~~including remote service method calls~~.

[Claim 18] (original) A system according to claim 1 may serialize the values of an object to a remote system and the receiving system may create a new instance of the object for pattern matching.

[Claim 19] (original) Objects according to claims 16 to 18 are considered temporal by the rule engine if the object's original instance and nodes reside on a remote system.

[Claim 20] (original) A system according to claim 1 defines three types of input channels: standard input, data distribution and node distribution.

[Claim 21] (currently amended) A system according to claim 1 defines a set of application interfaces and components to handle incoming data input, events and requests for pattern matching.

[Claim 22] (original) Input according to claim 21 is defined as standard input channel.

[Claim 23] (original) A system according to claim 1 defines a data distribution channel for sending and receiving remote data between rule engines.

[Claim 24] (currently amended) A system according to claim 1 defines a pattern distribution channel for distributing RETE nodes and patterns.

[Claim 25] (original) A system according to claim 1 considers an object as temporal if it was sent through the data distribution channels.

[Claim 26] (currently amended) Temporal objects according to claim 19 and 25 are used by the engine to perform pattern matching and ~~these objecsts are~~ may be discarded immediately after the pattern matching process is complete.

[Claim 27] (currently amended) A system according to claim 1 will route the index results of claim 26 back to the originating system using the data distribution channel.

[Claim 28] (original) A system according to claim 1 will update the index of the join and terminal nodes as a result of pattern matching according to claim 26.

[Claim 29] (original) A system according to claim 1 processes the outputs of the rule and produces a set of results, if the original event began locally.

[Claim 30] (original) A system according to claim 1 uses messaging system to route new events to a cluster of rule engines.

[Claim 31] (original) A messaging system according to claim 30 filters new messages and routes them to the correct engine.

[Claim 32] (original) A system according to claim 1 uses messaging system to route the final results as defined by a rule to the recipient.

[Claim 33] (original) A system according to claim 1 contains a component responsible for communicating with the messaging system.

[Claim 34] (original) A messaging component according to claim 33 is responsible for processing inbound events and generating new messages for outbound publication.

[Claim 35] (currently amended) A system according to claim 1 prefers to process new data events asynchronously using a messaging system, but is not required to use asynchronous communication.

[Claim 36] (original) Distributed nodes according to claim 14 contain information about the originating engine, a timestamp of when the nodes were distributed and a priority attribute.

[Claim 37] (original) The priority attribute according to claim 36 may be used by the engine to remove the nodes, if all local object instances have been removed from the working memory.

[Claim 38] (currently amended) Distributed nodes according to claim 14 will not be removed from the local pattern matching network, if data for those patterns is still being used, either in active rules about to fire or in remote rule engine instances ~~procedural attachment calls~~.

[Claim 39] (original) A system according to claim 1 may forward node distribution messages, if the system does not have sufficient resources.

[Claim 40] (original) A forward message according to claim 19 must retain the location of the originating engine and add the current engine's unique runtime name to a list of recipients.

[Claim 41] (original) A system according to claim 1 will notify the producer of the node distribution message of success or failure.

[Claim 42] (original) Distributed nodes according to claim 14 may be distributed at deployment time.

[Claim 43] (original) A system according to claim 1 will set an attribute of the input node to indicate that the pattern has been distributed.

[Claim 44] (original) An input node according to claim 43 will maintain a list of the remote systems and the total number of data objects routed remotely.

[Claim 45] (original) A system according to claim 1 may not attempt to distribute nodes that were distributed by another rule engine. Instead, it should notify the originating engine it cannot receive additional data until resources are available.

[Claim 46] (original) A system according to claim 1 may randomly select a remote engine to route data to, if the pattern is distributed to more than 1 engine.